Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Klemen Kastelic

Brezžično upravljanje robota preko spletnega vmesnika

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: prof. dr. Branko Šter

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:
Brezžično upravljanje robota preko spletnega vmesnika

Tematika naloge:

Razvijte spletni uporabniški vmesnik za nadzor nad robotom. Za strojni del projekta uporabite senzorje, ki bodo sporočali podatke o robotu. Cilj diplomskega dela je omogočiti enostavno uporabo robota preko spleta ali lokalnega omrežja.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Klemen Kastelic sem avtor diplomskega dela z naslovom:

Brezžično upravljanje robota preko spletnega vmesnika

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Branko Štera,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 21. december 2015 Podpis avtorja:

Zahvaljujem se svoji družini za podporo in financiranje pri študiranju. Posebej se zahvaljujem svojmu mentorju, prof. dr. Branko Šteru za nasvete in pomoč pri izdelavi diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	Hyper Text Markup Language	označevalni jezik
HTML5	Hyper Text Markup Language v5	nadgrajen označevalni jezik
CSS	Cascading Style Sheets	kaskadne stilske predloge
SASS	Syntactically Awesome Style Sheets	sintaktično super slogovne predloge
DOM	Document Object Model	objektni model dokumenta
URL	Uniform Resource Locator	enolični krajevnik vira
HTTP	HyperText Transfer Protocol	protokol za prenos hiperpovezav
ADC	Analog to Digital Converter	analogni v digitalni pretvornik
GPS	Global Positioning System	sistem globalnega pozicioniranja
SPI	Serial Peripheral Interface	serijski periferni vmesnik
IP	Internet Protocol	protokol za internetni naslov
USB	Universal Serial Bus	univerzalno serijsko vodilo
LiPo	Lithium Polymer battery	litijeva polimerna baterija
GPIO	General Purpose Input/Output	splošno namenski vhod/izhod
SSH	Secure Shell	varna lupina

Povzetek

V diplomskem delu smo razvili prototip robota, ki ga uporabnik lahko nadzira preko spletnega vmesnika, do katerega dostopa preko spletnega brskalnika na osebnem računalniku. Spletni vmesnik v realnem času sporoča stanje senzorjev in prenaša video s spletne kamere, ki je pritrjena na robota.

Zaledni del sistema poganja računalnik Raspberry Pi. Splošnonamenski vhodno-izhodni vmesnik komunicira z gonilnikom motorjev in s senzorji. Preko vtičnika USB povezana spletna kamera in brezžična kartica skrbita za video in brezžično komunikacijo.

V prvem delu opisujemo razvojna orodja in tehnologije, ki so bile uporabljene pri razvoju uporabniškega vmesnika, zalednega ter fizičnega dela. V drugem delu opisujemo idejo in načrtovanje uporabniškega vmesnika, zalednega in fizičnega dela. Tretji del opisuje funkcije in delovanje uporabniškega vmesnika, zalednega in fizičnega dela.

Ključne besede: javascript, node, socket.io, robot, GPS, ultrazvočni merilnik razdalje, spletna kamera, servo motor, Raspberry Pi, gonilnik motorjev.

Abstract

In this thesis we developed a prototype robot, which can be controlled by user via web interface and is accessible trough a web browser. Web interface updates sensor data and streams video captured with the web-cam mounted on the robot in real-time.

Raspberry Pi computer runs the back-end code of the thesis. General purpose input-output header on Raspberry Pi communicates with motor driver and sensors. Wireless dongle and web-cam connected trough USB, ensure wireless communication and video capture.

First part of the thesis describes the development tools and technologies that were used in development of the user interface, back-end and hardware part. Second part describes the idea and planning. Third part describes the functions it can do and development of the robot.

Keywords: javascript, node, socket.io, robot, GPS, ultrasonic range finder, web camera, servo motor, Raspberry Pi, motor driver.

Poglavje 1

Uvod

Spletne tehnologije se uporabljajo za razvoj čelnega in zalednega dela spletnih aplikacij. Nove spletne tehnologije se v zadnjih letih hitro razvijajo in se zaradi podpore na različnih operacijskih sistemih uporabljajo tudi za razvoj namiznih aplikacij. Spletni brskalniki so postali obvezna oprema vsake naprave, ki lahko dostopa do spleta. Večina operacijskih sistemov ima že standardno naložene spletne brskalnike, ki so preprosti za uporabo. Zato smo se odločili, da bomo za razvoj robota uporabili spletne tehnologije.

Robot je elektro-mehanski stroj, ki je voden s pomočjo računalniškega programa. Uporablja se za različne namene, npr. raziskovanje tujega sveta, pomoč pri reševanju iz nevarnega okolja, olajšanje vsakdanjih opravil, zdravstvo, zabavo in industrijo.

V okviru diplomskega dela se bomo ukvarjali z razvojem robota, ki ga nadzirajo uporabniki preko spletnega vmesnika. Robot bo uporabniku sporočal svojo lokacijo, stanje baterije, razdaljo od objekta spredaj in razdaljo od objekta zadaj. Preko brezžične povezave se bo v realnem času odzival na uporabniške ukaze in v živo predvajal video, ki ga zajema kamera.

Poglavje 2

Tehnologije in orodja

2.1 Razvoj uporabniškega vmesnika

Za razvoj so bile uporabljene različne tehnologije, ki so bile potrebne za uspešno delovanje uporabniškega vmesnika. Ločeno bomo opisali različne tehnologije ter razvojna orodja.

2.1.1 HTML in HTML5

HyperText Markup Language (HTML) [1] je standardiziran označevalni jezik, ki ga spletni brskalniki uporabljajo za razlago in sestavo vizualnih spletnih strani. HTML5 [2] je nadgradnja tehnologije HTML. Nadgradnja vsebuje podporo za najnovejše multimedijske vsebine, kot so video, avdio in vektorske slike.

HTML smo uporabili za izdelavo spletne strani, na kateri je uporabniški vmesnik. Na uporabniškem vmesniku prikazujemo ikono za stanje brezžične povezave, ki je sestavljena iz vektorske grafike, na katero lahko vplivamo s CSS lastnostmi. Ker HTML ne podpira vektorskih grafik, smo uporabili tudi HTML5. Uporaba vektorske grafike prihrani čas pri razvoju in odpravi uporabo statičnih slik, ki upočasnijo čas nalaganja spletne strani.

Koda iz uporabniškega vmesnika, ki prikazuje kvaliteto brezžične povezave:

```
<svg version="1.1"
    class="iconic iconic-signal iconic-signal-strong"
    xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    x="0px" y="0px"
    width="11.26px"
    height="8px"
    viewBox="0 0 11.26 8"
    enable-background="new 0 0 11.26 8"
    xml:space="preserve">
  <path class="iconic-signal-base"
        d="M6.337,7.247c
           -0.391-0.391-1.023-0.391-1.414,0l0
           .708,0.708L6.337,7.247z"/>
  <g class="iconic-signal-wave">
    <path class="iconic-signal-wave-inner"
          fill="none"
          stroke="#FFFFFF"
          stroke-miterlimit="10"
          d="M7.62,5.966c
             -1.098-1.098-2.88-1.098-3.977,0"/>
    <path class="iconic-signal-wave-middle"
          fill="none"
          stroke="#FFFFFF"
          stroke-miterlimit="10"
          d="M9.31,4.275C7
             .278,2.244,3.984,2.245,1.952,4.276"/>
    <path class="iconic-signal-wave-outer"
          fill="none"
          stroke="#FFFFFF"
          stroke-miterlimit="10"
```

```
28            d="M10.9,2.684c
                 -2.911-2.911-7.629-2.911-10.54,0"/>
29    </g>
30  </svg>
```

2.1.2 CSS in SASS

Cascading Style Sheets (CSS) [3] se uporablja za oblikovanje vizualno privlačnih spletnih strani, uporabniških vmesnikov za spletne aplikacije in mnogih mobilnih aplikacij. *Syntactically Awesome Stylesheets* SASS [4] je skriptni jezik, ki je bil razvit zaradi kompleksnejših spletnih strani in boljše organizacije jezika CSS. Prevajalnik prevede SASS kodo v navaden CSS, ki je standardno znan vsem spletnim brskalnikom. SASS nudi uporabo spremenljivk, gnezdenje, različne metode, ki podpirajo argumente, zanke, dedovanje, matematične operacije ter uvažanje drugih SASS datotek, ki nudijo razvijalcem lažji in hitrejši razvoj spletnih strani in uporabniških vmesnikov.

Uporabili smo SASS, ki ga s prevajalnikom prevedemo v CSS. SASS je za razvoj hitrejši, zaradi bolj pregledne kode in dodatnih funkcij, ki jih nudi. Največ smo uporabili spremenljivke, gnezdenje in dedovanje. Odsek kode spodaj prikazuje uporabo dedovanja, spremenljivk in gnezdenja za oblikovanje prikaza podatkov, ki jih dobimo iz ultrazvočnih merilcev razdalje:

```
1  $sensor-width: 100px;
2  $sensor-height: 18px;
3
4  .sensors{
5    .sensor{
6      text-align: center;
7      position: absolute;
8      width: $sensor-width;
9      height: $sensor-height;
10     line-height: $sensor-height;
```

```
11      margin: 5px;
12      font-size: 11px;
13      text-shadow: 0px 0px 4px black;
14    }
15
16    .sensor-front{
17      @extend .sensor;
18      margin-left: -($sensor-width/2);
19      top: 0px;
20      left: 50%;
21    }
22
23    .sensor-back{
24      @extend .sensor;
25      margin-left: -($sensor-width/2);
26      bottom: 0px;
27      left: 50%;
28    }
29  }
```

2.1.3 JavaScript

JavaScript [5] je objektni skriptni programski jezik, ki ga je leta 1995 razvilo podjetje Netscape, za idelavo interaktivnih spletnih strani. Glavni spletni brskalniki, kot so Google Chrome, Firefox, Safari in Opera, podpirajo JavaScript brez posebnih vtičnikov. Najpogosteje je JavaScript uporabljen na uporabnikovi strani spletne strani za asinhrono delovanje in spreminjanje vsebine dokumenta. Uporablja se tudi na zaledni strani z razvojnimi okolji kot je Node.js, pri razvoju iger ter ustvarjanju namiznih in mobilnih aplikacij.

Zaradi zahteve po manipulaciji elementov na uporabnišekm vmesniku smo uporabili JavaScript. Za zajem dogotkov tipkovnice, miške in zaslona na dotik skrbi JavaScript, ki potem ustrezno izvede zahtevo. Primer zajema

tipkovnice z skriptnim programskim jezikom JavaScript na uporabniškem vmesniku, ki preveri, če je bila pritisjena tipka "Enter":

```
1 document.addEventListener("keydown", keyDownTextField
    , false);
2
3 function keyDownTextField(e) {
4   if(e.keyCode==13) {
5     alert("You hit the enter");
6   } else {
7     alert("You didn't hit enter");
8   }
9 }
```

2.1.4 jQuery

Za lažje pisanje skript in *Document Object Model* (DOM) manipulacijo za HTML je bila leta 2006 ustvarjena knjižnica jQuery [7]. Velja za eno izmed najbolj priljubljenih JavaScript knjižnic na svetu, ki je uporabljena na veliko največjih spletnih straneh. Ustvarjena je za lažjo navigacijo po dokumentu, iskanje DOM elementov, kreiranje animacij, upravljanje z dogodki ter izvajanje asinhronih funkcij.

Knjižnico jQuery smo uporabili, ker zmanjša količino kode za naloge kot so spreminjanje besedila, premikanje elementov po strani in izvajanje animacij. Uporabljamo jo pri osveževanju informacij na uporabniškem vmesniku. Primer primerjave med jQuery in JavaScript funkcijo "fadeIn()" je prikazan spodaj:

```
1 // jQuery
2 $("#front-senzor").fadeIn();
3
4 // JavaScript
5 function fadeIn(el){
```

```
6     el.style.opacity = 0;
7     el.style.display = "block";
8
9     (function fade() {
10      var val = parseFloat(el.style.opacity);
11      if (!((val += .1) > 1)) {
12        el.style.opacity = val;
13        requestAnimationFrame(fade);
14      }
15    })();
16  }
17  var el = document.getElementById("front-senzor");
18  fadeIn(el);
```

2.1.5 Socket.io

Socket.io [6] je Node.js modul za spletne aplikacije v realnem času. Omogoča obojestransko komunikacijo v realnem času med uporabniškim vmesnikom ter zalednim delom spletne aplikacije. Sestavljen je iz knjižnice na uporabniški strani, ki se izvaja v spletnem brskalniku, ter knjižnice za zaledno stran, ki jo izvaja Node.js. Obe knjižnici imata identični vmesnik za programiranje.

Potrebovali smo komunikacijo v realnem času, zato smo uporabili knjižnico Socket.io. Knjižnico Socket.io uporabljamo pri komunikaciji zalednega dela z uporabniškim vmesnikom in obratno. Primer kode prikazuje enostavno sporočilo, ki ga pošlje zaledni del, ko uporabniški vmesnik vzpostavi povezavo. Uporabniški vmesnik prikaže opozorilno okno, ki izpiše "Hello world!":

```
1  // Uporabniski vmesnik
2  var socket = io();
3  socket.on('message', function (msg) {
4      alert(msg.data);
```

```
5 });
```

```
1 // Zaledni del
2 var express = require('express');
3 var app = express();
4 var http = require('http')(app);
5 var io = require('socket.io')(http);
6
7 io.on('connection', function(socket){
8   io.emit('message', { data: "Hello world!" });
9 });
```

2.1.6 Google Maps

Google Maps [8] storitev podjetja Google, ki vsebuje satelitske posnetke, zemljevide ulic in panoramski pogled ulic. Z brezplačnim programskim vmesnikom Google Maps API lahko vgradimo Google Maps zemljevide v uporabniški vmesnik, kjer lahko vplivamo na njihov izgled in delovanje.

Hoteli smo prikazati lokacijo robota na zemljevidu, zato smo uporabili programski vmesnik Google Maps API. Enostavnost uporabe prikazuje funkcija *initializeMap*() spodaj, ki se kliče pri inicilizaciji. Funkcija *initilizeMap*() v pravilen DOM element ustvari zemljevid, ki ima določene začetne nastavitve. Na zemljevid postavi tudi marker, ki prikazuje lokacijo robota. Zemljevid je na uporabniškem vmesniku skrit, če nam GPS modul ne vrne trenutne lokacije.

```
1 var map, marker;
2 function initializeMap() {
3   var pos = { lat: 46.0500176, lng: 14.4668417 };
4
5   var mapOptions = {
6     zoom: 18,
7     center: pos,
```

```
8       disableDefaultUI: true
9     };
10
11    map = new google.maps.Map($('#map-canvas'),
          mapOptions);
12    marker = new google.maps.Marker({
13      position: pos,
14      map: map
15    });
16  }
```

2.2 Zaledni sistem

To poglavje našteva in opisuje tehnologije, ki so bile uporabljene za razvoj zalednega dela diplomske naloge. Zaledni del poganja računalnik Raspberry Pi, ki preko povezave Socket.io komunicira z uporabniškim vmesnikom.

2.2.1 Node.js in NPM

Node.js [9] je odprtokodna aplikacija, ki uporablja dogodkovno usmerjen, neblokirajoč vhodno-izhodni model. Deluje samo na eni niti, vendar z uporabo neblokirajočih vhodnih-izhodnih klicev lahko podpira več deset tisoč sočasnih povezav. Upravljalec s paketi *Node Package Manager* (NPM) [11] je orodje, ki omogoča preprosto uporabo Node.js modulov in je del Node.js aplikacije.

Node.js nam omogoča, da zaledni del sistema pišemo v skriptnem programskem jeziku JavaScript. Za Node.js smo se odločili, ker ima veliko modulov, ki so nam pomagali pri razvoju zalednega dela robota. Module, ki smo jih uporabili, bomo opisali v naslednjih poglavjih.

2.2.2 Express

V okviru diplomskega dela smo potrebovali orodje za serviranje datotek spletnim brskalnikom. Odločili smo se za modul Express, ki smo ga uporabili

za enostaven *Hypertext Transfer Protocol* (HTTP) strežnik. Modul Express smo v projekt dodali z upravljalcem z moduli NPM. Odrezek kode diplomske naloge prikazuje ustvarjanje enostavnega HTTP strežnika, ki posluša na vratih 80. Ko se uporabnik z spletnim brskalnikom poveže na mu servira datoteko */client/index.html*.

```
1  var express = require('express');
2  var app = express();
3  var http = require('http')(app);
4
5  app.use(express.static(__dirname + '/client'));
6  app.get('/', function(req, res){
7    res.sendFile(__dirname + '/client/index.html');
8  });
9
10 http.listen(80, function(){
11   console.log('listening on ' + ip +':80');
12 });
```

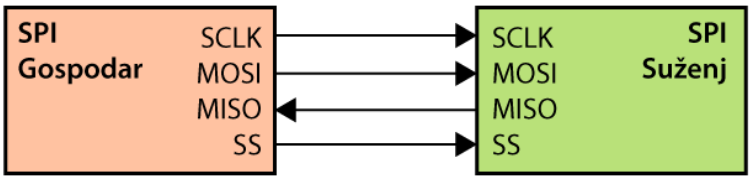
2.2.3 Univerzalni asinhroni serijski vmesnik

Universal Asynchronous Receiver/Transmitter (UART) [13] je integrirano vezje, ki skrbi za serijsko komunikacijo. UART na Raspberry Pi komunicira preko dveh pinov, ki so na voljo na *General-purpose input/output* (GPIO) vmesniku, to sta: signal za oddajanje podatkov (TX) in signal za sprejemanje podatkov (RX). UART vmesnik komunicira na zaporeden način. Upošteva različne nastavitve kot so število bitov, hitrost prenosa, končni bit in paritetni bit. Število bitov za vsak znak je lahko 5, 6, 7, 8 ali 9. Najpogosteje se uporablja 8-bitni zapis, 5 in 7 pa ima običajno smisel samo na starejši opremi. Na koncu vsakega znaka se pošilja tudi bit za oznako konca prenosa, kar pripomore pri sinhronizaciji s tokom znakov. Bit parnosti se uporablja za odkrivanje napak pri prenosu in se tako kot končni bit postavi na konec bitov za znak, ki sporoči, če je število bitov v nizu z vrednostjo ena liho ali

sodo. Podprte bitne hitrosti so 75, 110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600 in 115200 bitov/s.

UART smo uporabili pri komunikaciji med računalnikom Raspberry Pi in GPS modulom. Zaradi lažje uporabe UART vmesnika smo uporabili Node.js modul "serialport", ki skrbi za branje podatkov. Preko povezave Socket.io nato te podatke pošljemo na uporabniški vmesnik. Koda enostavnega primera branja iz UART vmesnika:

```
1  var serialport = require("serialport");
2  var serialPort = new SerialPort("/dev/ttyAMA0", {
3    baudrate: 9600,
4    parser: serialport.parsers.readline("\n")
5  }, false);
6
7
8  var SERIAL = {
9    openSerialPort: function(callback){
10     serialPort.open(function (error) {
11       if(!error){
12         serialPort.on('data', function(data) {
13             // Returns collected data
14             alert(data);
15         });
16       } else {
17         // Error has occurred.
18         alert("Error");
19       }
20     });
21   }
22 };
23
24 module.exports = SERIAL;
```



Slika 2.1: Primer vodila SPI.

2.2.4 SPI

Serial Peripheral Interface (SPI) [14], sinhroni serijski komunikacijski vmesnik, smo potrebovali pri komunikaciji računalnika Raspberry Pi z analogno-digitalnim pretvornikom MCP3008, ki smo ga uporabili za branje stanja baterije. SPI uporablja arhitekturo gospodar-suženj s samo enim gospodarjem. Gospodar lahko izbira posameznega sužnja, s katerim bo komuniciral. Slika 2.1 prikazuje primer SPI vodila.

Vodilo določa štiri ločene signale:

- SCLK: serijska ura,
- MOSI: izhod gospodarja, vhod sužnja,
- MISO: vhod gospodarja, izhod sužnja,
- SS: izbira sužnja

Z uporabo "spi" modula, ki smo ga enostavno dodali zalednemu delu preko upravljalca s paketi NPM, smo lahko komunicirali s MCP3008 čipom. Primer uporabe "spi" modula je napisan spodaj.

```
1 var spi = new SPI.Spi('/dev/spidev0.0');
2
3 var buf1 = new Buffer(8);
4 spi.read(buf1, function(device, buf2) {
5   var s = "";
6   for (var i=0; i < buf2.length; i++)
```

```
7       s = s + buf2[i] + " ";
8     });
9     console.log(s);
10 )};
```

2.2.5 Pi-blaster

Pi-blaster [15] je program, ki deluje kot proces v ozadju operacijskega sistema, ki ga poganja računalnik Raspberry Pi in nudi vmesnik za nadzor nad več pulzno-širinskih modulacij na izhodih GPIO.

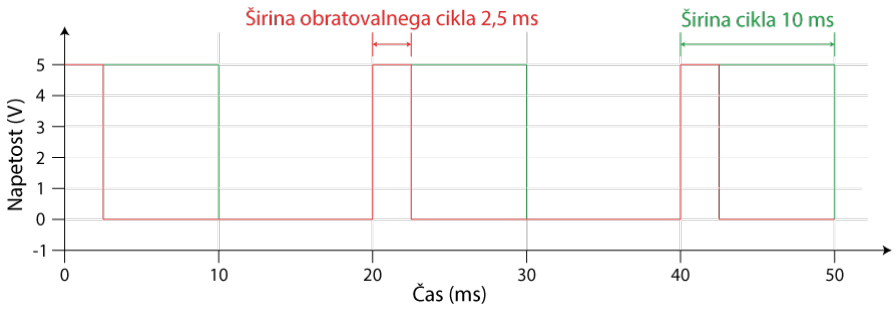
Pulse Width Modulation (PWM) [16] je metoda za generiranje analognega signala z uporabo digitalnega vira. Način modulacije določata obratovalni cikel in frekvenca delovanja. Frekvenca delovanja določa čas, v katerem PWM opravi en cikel. Programa Pi-blaster deluje s frekvenco 100Hz, kar pomeni, da je en cikel dolg 10 ms. Obratovalni cikel določa odstotek časa enega cikla, kjer je signal v visokem stanju. Program Pi-blaster ustvari datoteko, ki je vmesnik za gonilnik naprave. Z ukazom v ukazni vrstici

```
1 echo "3=0.25" > /dev/pi-blaster
```

lahko nastavimo izhod pina številka 3 na PWM obratovalni cikel 25%. Slika 2.2 prikazuje primer PWM obratovalnega cikla 25%. Program pošilja PWM pulze, dokler mu ne sporočimo drugače.

Potrebovali smo vmesnik, ki bo lahko pošiljal ukaze v “/dev/pi-blaster” datoteko iz Node.js zalednega dela. Zato smo ustvarili modul servo, ki pošilja ukaze in skrbi za delovanje servo motorjev. Izrezek iz modula servo spodaj skrbi za pisanje ukazov v datoteko ”/dev/pi-blaster”. Node.js že vsebuje modul ”fs”, ki se uporablja za branje in pisanje datotek.

```
1 var fs = require('fs');
2
3 var SERVO = {
```



Slika 2.2: Primer obratovalnega cikla 25%.

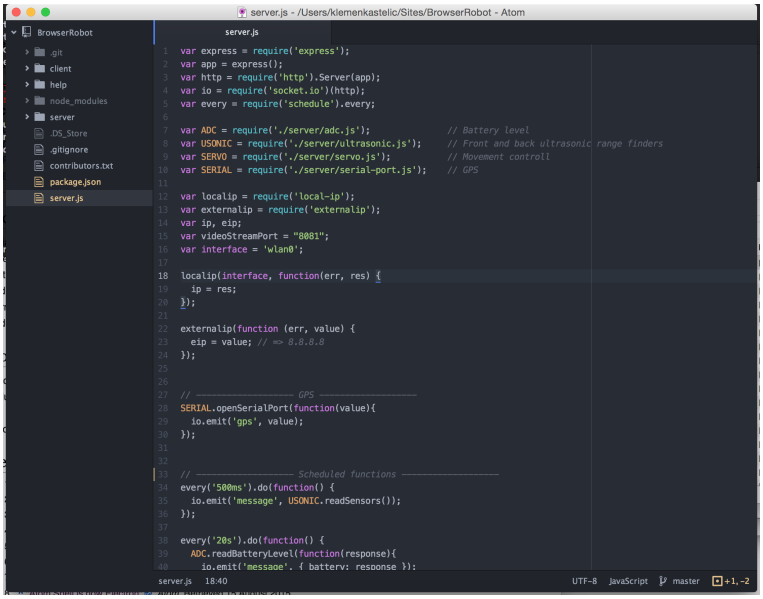
```
4   writeCommand(pinNumber, value) {
5     var buffer = new Buffer(pinNumber + "=" + value +
          "\n");
6     var fd = fs.open("/dev/pi-blaster", "w",
         undefined, function(err, fd) {
7       if (!err)
8         fs.write(fd, buffer, 0, buffer.length, -1,
             function(error, written, buffer) {
9           if (!error) fs.close(fd);
10        });
11      }
12    });
13  }
14};
15
16module.exports = SERVO;
```

2.2.6 Schedule

Schedule [17] je knjižnica za upravljanje s časovnimi nalogami, ki jo izvaja Node.js. Z njo lahko naredimo časovno izvedene funkcije z določenim časovnim intervalom, ki se izvaja neskončno ali pa do prekinitve. Primer funkcije, ki se izvede vsake 3 sekunde:

```
every('3s').do(function() {
    // do something
});
```

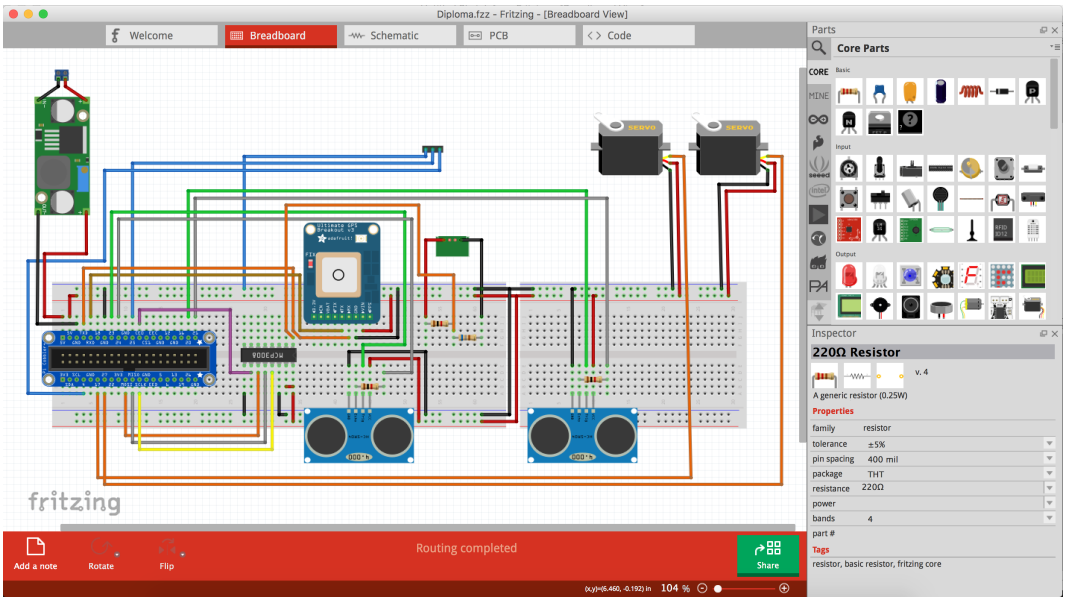
Potrebovali smo knjižnico, ki nam bo izvajala določene funkcije v izbranih intervalih. Uporabili smo jo za osveževanje stanja baterije, kvalitete brezžične povezave in razdalje ultrazvočnih merilnikov.



Slika 2.3: Razvojno okolje Atom.

2.3 Razvojno okolje

Za razvoj smo uporabili Atom [18], brezplačen odprtokodni urejevalnik besedila in izvorne kode. Prednost tega urejevalnika je, da ima veliko podpore razvijalske skupnosti. Podpira vtičnike napisane v Node.js in ima vgrajen nadzor nad repozitorijem Git. S pomočjo vtičnikov podpira poudarjanje sintakse skoraj vsakega jezika.



Slika 2.4: Orodje Fritzing.

2.4 Fritzing

Za pomoč pri načrtovanju elektronskega vezja smo uporabili Fritzing [19], ki je odprtokodno orodje za načrtovanje elektronskih vezij. Z njim lahko iz grafičnega pogleda rišemo elektronske sheme in načrtujemo tiskano vezje. Uporabili smo ga za izris in dokumentiranje elektronskega vezja. Grafična in elektronska shema nam je pomagala pri predstavi povezav senzorjev na GPIO vmesniku.

Poglavje 3

Strojna oprema

3.1 Raspberry Pi

Raspberry Pi [20] je cenovno ugoden računalnik v velikosti kreditne kartice, ki lahko preko mrežne povezave ali GPIO vmesnika komunicira z zunanjim svetom. Specifikacije računalnika Raspberry Pi so opisane v tabeli 3.1. Računalnik Raspberry Pi smo uporabili, ker je cenovno ugoden in majhen. Operacijski sistem Linux, ki ga poganja računalnik Raspberry Pi, nam je omogočil poganjanje zalednega dela na robotu.

3.2 Ultrazvočni merilnik razdalje

Ker smo hoteli na uporabniškem vmesniku prikazati odaljenost objektov od robota smo za zaznavanje razdalje uporabili modul HC-SR04 [21], ki uporablja ultrazvok za določitev oddaljenosti objekta od senzorja. Izmeri lahko razdalje od 2 cm do 500 cm z natančnostjo 3 mm.

Ultrazvočni merilnik razdalje deluje tako, da odda kratek ultrazvočni pulz, ki se odbija od objektov. Nato prejme ta pulz in ga pretvori v električni signal. Naslednji pulz se lahko odda, ko odbojni pulz zbledi. To časovno obdobje se imenuje ciklično obdobje, ki ne sme biti manjše kot 50

Cena	35 EUR
CPE	900 MHz quad-core ARM Cortex-A7
Delovni spomin	1 GB
USB 2.0	4
Zvok	3.5mm in HDMI
Poraba energije	800 mA (4.0 W)
Dimenzije	85.60 mm × 56.5 mm
Masa	45 g

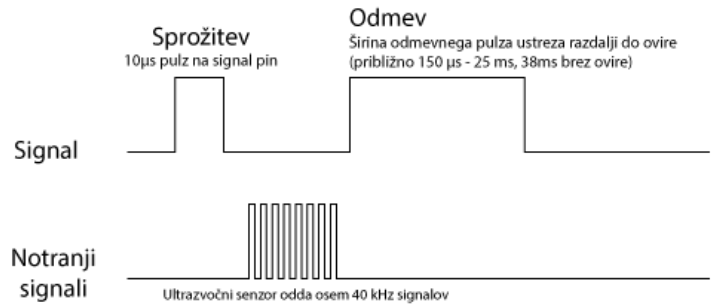
Tabela 3.1: Specifikacije računalnika Raspberry Pi.

ms. Če se 10 µs pulz pošlje na sprožilni vhod, bo modul oddal osem 40 kHz ultrazvočnih signalov in zaznal odboj. Če odboj ni zaznan bo, izhodni signal dolg 38 ms. Izmerjena razdalja do ovire je sorazmerna širini odmevnega pulza in se lahko izračuna po spodnji enačbi, kjer je x širina odmevnega pulza v milisekundah.

$$razdalja(x) = \frac{x}{58\frac{cm}{ms}}$$

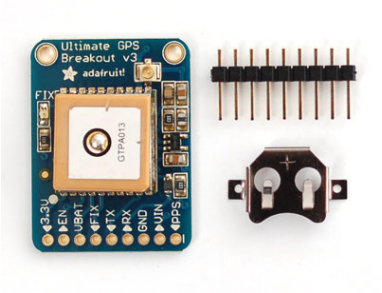
Modul s pomočjo knjižnice r-pi-usonic [22] komunicira preko GPIO vmesnika. Knjižnica deluje tako, da na signal pin pošlje 10 µs pulz, nato čaka na odmevni pulz. Širino odmevnega pulza knjižnica meri v milisekundah, ki ga nato deli z 58 cm/ms, da dobimo razdaljo v centimetrih. Zvok potuje s hitrostjo 29 cm/ms, vendar odmevni signal potuje od senzorja do objekta in nazaj, zato množimo hitrost zvoka z 2 in dobimo 58 cm/ms. Razdaljo do objekta potem zaledni del pošlje preko Socket.io povezave na uporabniški vmesnik. Knjižnica jQuery pa poskrbi, da se razdalje izpišejo na spletnem vmesniku.

```
1 var usonic = require('r-pi-usonic');
2
```



Slika 3.1: Časovni diagram poteka meritve razdalje.

```
3  // "echoPin":24,"triggerPin":23,"timeout":750,"delay
       ":60,"rate":5
4  var backSensor = usonic.createSensor(24, 23, 500);
5  var frontSensor = usonic.createSensor(21, 20, 500);
6
7  var USONIC = {
8    readFrontSensor: function(){
9      return Math.round(frontSensor());
10   },
11
12   readBackSensor: function(){
13     return Math.round(backSensor());
14   },
15
16   readSensors: function(channel, callback){
17     return {
18       front: USONIC.readFrontSensor(),
19       back: USONIC.readBackSensor()
20     };
21   }
22 };
23
```



Slika 3.2: Adafruit Ultimate GPS modul [24].

```
24 module.exports = USONIC;
```

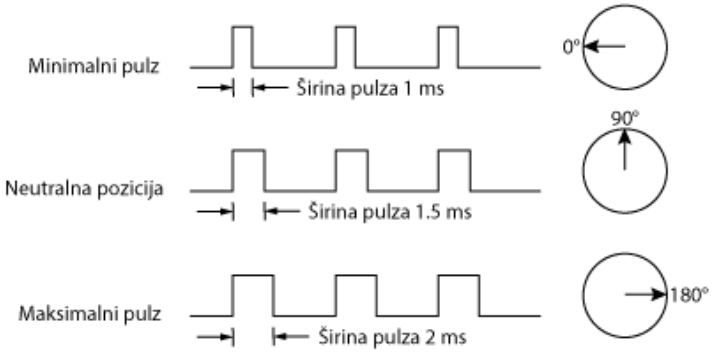
Časovna funkcija vsakih 500 ms prebere vrednosti, ki jih vrne ultrazvočni merilnik razdalje in jih pošlje preko Socket.io poveave na uporabniški vmesnik.

```
1 every('500ms').do(function() {
2   // Read and emit ultrasonic range finders data
3   io.emit('message', USONIC.readSensors());
4 });
```

3.3 GPS modul

Z namenom zaznavanja robotove lokacije smo uporabili Adafruit Ultimate GPS [24] modul z vgrajeno anteno, ki je cenovno ugoden in ima lahko dostopne pin za prototipiranje. Modul pošilja *National Marine Electronics Association* (NMEA) [37] stavke preko UART vmesnika v zaledni del vsako sekundo. Zaledni del potem te podatke pošlje uporabniškemu vmesniku preko Socket.io povezave. NMEA je kombinirana električna in podatkovna specifikacija za komunikacijo med pomorsko elektroniko kot so sonarji, žirokompasi, avtopiloti, *Global Positioning System* (GPS) [23] sprejemniki ter veliko drugih vrst inštrumentov.

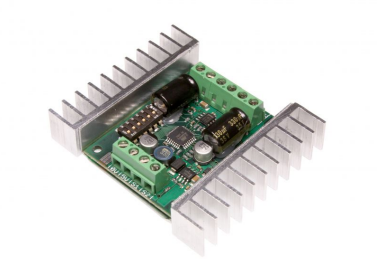
Slika 3.3: Tipičen servo motor [26].



Slika 3.4: Primeri delovanja servo motorja.

3.4 Servo motor

Servo motor [25] je motor, ki se uporablja za natančno krmiljenje položaja, hitrosti in pospeška izhodne gredi. Sestavljen je iz motorja in potenciometra za povratno informacijo o položaju, ki ju povezuje krmilno elektronsko vezje. Položaj izhodne gredi servo motorja upravljamo s pulzno-širinsko modulacijo. Pulz širine 1-milisekundnde sporoča servo motorju, naj se postavi na 0 stopinj, 1.5-milisekund na 90 stopinj in 2-milisekundi na 180 stopinj. Pulzi se pošiljajo vsakih 10 milisekund. Slika 3.4 prikazuje opisan primer. Ker smo želeli, da ima uporabnik nadzor nad položajem spletne kamere, smo uporabili dva servo motorja za nagib in obračanje spletne kamere.



Slika 3.5: Gonilnik motorjev Sabertooth 2x15A [28].

3.5 Spletna kamera

Logitech visoko ločljivostna spletna kamera C270 [27] je USB spletna kamera, ki lahko zajema 1280 pik širok in 720 pik visok video. Kamera se z USB kablom poveže na računalnik Raspberry Pi, ki ima privzeto naložene gonilnike na operacijskem sistemu. Uporabili smo spletno kamero Logitech C270, ker je privzeto podprta na operacijskem sistemu in ne potrebuje dodatnega napajanja.

3.6 Gonilnik motorjev

Kontroliranje in poganjanje motorjev je pomemben del našega dela, da se robot lahko premika po prostoru. Odločili smo se, da izberemo gonilnik motorjev Sabertooth [28], ki podpira upravljanje dveh motorjev. Uporabili smo ga zato, ker nam je bil na voljo iz osebnih projektov. Podpira analogno krmiljenje, radijske krmilnike, serijski vhod, serijski vhod s paketi in mikrokrmilniške pulze. Robot ima dva motorja na levi in dva motorja na desni strani mobilne platforme, ki sta povezana, da delujeta kot en motor.

Slika 3.6: Mobilna platforma Dagu Wild Trumper 4WD [29].

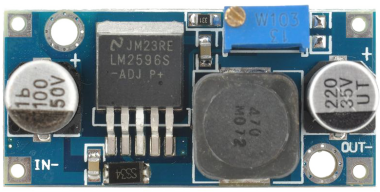
3.7 Mobilna platforma

Hoteli smo mobilno platformo, ki se bo dobro odnesla tudi v zunanjem okolju in bomo lahko na njo pritrdili vse komponente. Zato smo se odločili za mobilno platformo Dagu Wild Thumper 4WD [29], ki združi vse strojne komponente skupaj, jih varuje pred zunanjim okolju in ima že vgrajene motorje z kolesi. Z vgrajenim vzmetenjem in velikimi kolesi je primerna za bolj zahtevna zunanja okolja kot so makadam, travnik ali gozd. Šasija je iz aluminija s 4 mm luknjami s pomočjo katerih smo pritrdili servo motorje za obračanje spletne kamere, ultrazvočna senzorja in gonilnik motorjev. Ima že vgrajene motorje z zobniki, ki izboljšajo navor pogonu. S tem preprečijo poškodbe in omogočijo boljše delovanje motorjev. Najvišja hitrost robota je približno 7 km/h. Slaba stran te mobilne platforme je visoka cena.

3.8 Napajanje

3.8.1 Baterija

Uporabili smo polnilno baterijo ZIPPY 5000mAh [30] *Lithium polymer battery* (LiPo) [31], ki je lahka, ima visoko energijsko kapaciteto in visoko stopnjo praznjenja. NSlaba lastnost LiPo baterij je visoka cena, kratka doba



Slika 3.7: Regulator napetosti LM2596S 20083.

delovanja (okoli 300 do 500 polnilnih ciklov) in nevarnost vžiga pri poškodbi baterijskih celic ali napačnem polnjenju.

3.8.2 Regulator napetosti

Ker je izhodna napetost popolnoma napolnjene baterije 12.5 voltov, jo moramo za uporabo z Raspberry Pi znižati na 5 voltov. Uporabili smo elektronsko vezje LM2596S 20083 z regulatorjem napetosti, ki lahko pretvori od 3 do 40 voltov v 1.5 do 35 voltov. Odločili smo se za elektronsko vezje LM2596S 20083, ker je cenovno ugodno, majhno in podpira širok razpon vhodne in izhodne napetosti. Z obračanjem majhnega vijaka na potenciometru, ki je na sliki 3.7 elektronskega vezja zgoraj desno, se lahko nastavi želena izhodna napetost.

3.9 Analogno-digitalni pretvornik

Raspberry Pi nima analognega vhoda, zato smo morali uporabiti vezje, ki pretvori analogno napetost v digitalni zapis. MCP3008 [32] je čip, ki komunicira preko vmesnika SPI in ima 8 analognih vhodnih kanalov. Za čip MCP3008, smo se odločili zaradi nizke cene in želje po analognih vhodih. Analogni vhodni kanali sprejmejo od 5 voltov do 0 voltov, ki jih potem pretvori v digitalno obliko. Digitalna oblika zapisa je predstavljena s števili od 1025 do 0, kjer število 1025 predstavlja 5 V in število 0 predstavlja 0 V.

Poglavje 4

Načrtovanje

4.1 Ideja

Odločili smo se, da bomo razvili robota, ki ga bo uporabnik lahko upravljal preko spletnega vmesnika. Na spletnemu vmesniku bo uporabnik lahko v realnem času spremljal video prenos iz spletne kamere, ki bo na robotu, GPS lokacijo robota, kvaliteto brezžične povezave, stanje baterije, in razdaljo od ovir spredaj in zadaj. Spletno kamero bo uporabnik lahko premikal za boljšo predstavo prostora, v katerem se nahaja. Za komunikacjo z zunanjim svetom pa bo lahko uporabnik tudi posnel zvočno sporočilo, ki ga bo robot potem predvajal.

4.1.1 Zahtevane in zaželene funkcionalnosti

Glede na opis ideje podajamo specifikacije funkcionalnosti robota in uporabniškega vmesnika:

- Nadzor nad pogonom robota
- Prenos videa v živo
- Obračanje kamere za boljšo predstavo okolja
- Senzorji za merjenje razdalje od ovire

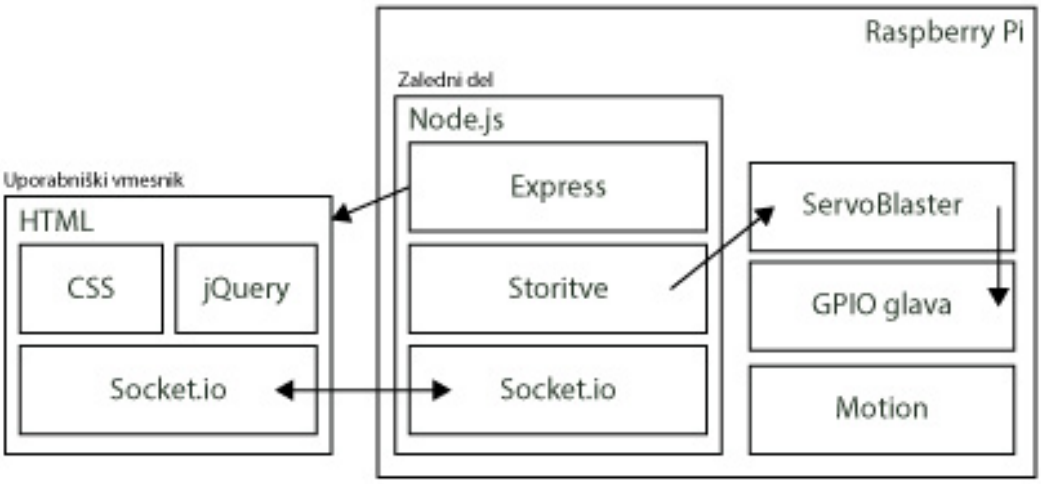
- GPS lokacija
- Stanje baterije
- Zvočna komunikacija

4.2 Izbira platforme

Izbrali smo računalnik Raspberry Pi, ki poganja operacijski sistem Linux in lahko preko Ethernet vtilnika, USB vtičnikov in GPIO vmesnik komunicira z zunanjimi komponentami. Računalnik Raspberry Pi zmore zajemati video in poganjanjati svoj strežnika v enem. Preizkusili smo tudi računalnik BeagleBone, vendar smo zaradi višje cene, podpore samo enega USB vtičnika in slabše procesorske moči izbrali računalnik Raspberry Pi.

4.3 Arhitektura

Raspberry Pi deluje kot strežnik in krmilinik za robota. Express knjižnica posreduje HTML, ki potem pridobi še JavaScript in CSS. Komunikacija uporabniškega vmesnika in zalednega dela poteka preko Socket.io, ki potem kliče storitve. Storitve nadzirajo delovanje motorjev, senzorjev, GPS modula in branje stanja baterije.



Slika 4.1: Arhitektura sistema.

Poglavje 5

Razvoj

5.1 Vzpostavitev sistema

5.1.1 Operacijski sistem

Uporabili smo operacijski sistem Raspian [35], ki temelji na okolju Debian, in je optimiziran za strojno opremo Raspberry Pi. Sistemsko sliko je potrebno namestiti na spominsko kartico, ki se nato vstavi v računalnik Raspberry Pi. Ker smo uporabili serijska vrata in SPI, smo morali v nastavitvenem oknu, ki je dostopno z ukazom `sudo rasp-config`, te module vklopiti.

5.1.2 Protokol FTP

Za dostop do datotek projekta, ki so shranjene na spominski kartici računalnika Raspberry Pi, smo uporabili *File Transfer Protocol* (FTP), ki je protokol za prenos datotek med računalniki z različnimi operacijskimi sistemi. Z uporabo protokola FTP je bil razvoj hitrejši od prenosa skozi Git repozitorij, ki bi za vsako spremembo potreboval potisk kode na repozitorij in nato prednost teh sprememb na Raspberry Pi. Operacijski sistem Raspian privzeto podpira FTP, zato ga je zlahka uporabiti. Geslo in uporabniško ime sta ista kot za prijavo v sistem ali preko varne lupine. Nismo uporabili protokol *Secure File Transfer Protocol* SFTP, ker smo datoteke prenašali samo v lokalnem

omrežju..

5.1.3 Motion

Motion [36] je program, ki spremlja video signal ene ali več kamer in je zmožen zaznati spremembe na sliki, oz. lahko zazna gibanje. Napisan je v programskem jeziku C za operacijski sistem Linux. Deluje tako, da prenese sliko vedno, ko zazna gibanje. Zaradi zahteve po prenosu v realnem času smo kvaliteto videa znižali na 352×288 pik. Program Motion se ob zagonu samodejno zažene kot proces v ozadju in uporablja 10% - 12% procesorske moči in delovnega spomina za delovanje. Zamik videa v živo je 30 ms - 60 ms.

5.1.4 Zagonska skripta

Ker se skripta Node.js ne izvede samodejno ob zagonu, je bilo potrebno napisati zagonsko skripto, ki je shranjena v "/etc/init.d/"direktoriju. Obstaja več načinov, da zaženemo skripto Node.js. Izbrali smo orodje Forever in Linux zagonsko skripto. Forever je vmesnik za ukazno vrstico, ki skrbi, da skripte delujejo neprekinjeno. Z zagonsko skripto nato zaženemo ali ustavimo orodje Forever, ki skrbi za neprekinjeno delovanje našega Node.js zalednega dela.

5.2 Uporabniški vmesnik

5.2.1 Komunikacija z zalednim sistemom

Komunikacija poteka skozi knjižnico Socket.io, ki je na uporabniški in zaledni strani. Uporabniki, ki so trenutno povezani, vplivajo na delovanje robota. Odzivnost robota je s pomočjo Socket.io povezave v realnem času tudi preko spleta. Uporabniški vmesnik posluša na kanalih za spremembe GPS lokacije, baterije, ultrazvočnih senzorjev, pritisnjenih gumbov, povezavo novega

uporabnika in prekinitev povezave uporabnika. Sporoča samo spremembo pritisnjenih gumbov.

5.2.2 Zajemanje vhodnih naprav

S pomočjo knjižnice jQuery zajemamo dogodek, ko uporabnik pritisne tipko na tipkovnici, ter dogodek, ko jo spusti. Uporabnik lahko gumb na zaslonu pritisne tudi z miško ali na zaslonu na dotik. Podatki o pritisnjenih gumbih so hranjeni v JSON objektu, ki se pošlje na zaledni del ob vsaki spremembi. Vsako spremembo zaledni del pošlje nazaj vsem povezanim uporabnikom, ki lahko spremljajo upravljanje robota.
Prikazan del JavaScript kode na uporabniškem vmesniku nam omogoča, da zajemamo dogodke tipkovnice, miške in zaslona na dotik. Nato s funkcijo "handleKey"pošljemo ustrezne podatke na zaledni del preko povezave Socket.io. Gumbi na uporabniškem vmesniku imajo svoj identifikator, preko katerega vemo, kateri gumb je bil pritisjen.

```
1  var pressedBtnId, oldKey;
2
3  $(document).keydown(function( event ) {
4    if(oldKey != event.which){
5      handleKey(event.which, true);
6      oldKey = event.which;
7    }
8  });
9
10 $(document).keyup(function( event ) {
11   handleKey(event.which, false);
12   oldKey = null;
13 });
14
15 $(".key").on('touchstart', function(e){
16     var id = $(this).attr('id');
17     handleKey(idToKeyCode(id), true);
```

```
18 })
19
20 $(".key").on('touchend', function(e){
21     var id = $(this).attr('id');
22     handleKey(idToKeyCode(id), false);
23 })
24
25 $(".key").mousedown(function(e){
26   var id = $(this).attr('id');
27   pressedBtnId = id;
28   handleKey(idToKeyCode(id), true);
29 });
30
31 $(".key").mouseup(function(e){
32   var id = $(this).attr('id');
33   handleKey(idToKeyCode(id), false);
34 });
35
36 $(window).mouseup(function() {
37   handleKey(idToKeyCode(pressedBtnId), false);
38 });
39
40 function idToKeyCode(id){
41   if(id == "forward") return 38;
42   if(id == "backward") return 40;
43   if(id == "left") return 37;
44   if(id == "right") return 39;
45
46   if(id == "a") return 65;
47   if(id == "d") return 68;
48   if(id == "e") return 69;
49   if(id == "f") return 70;
50   if(id == "q") return 81;
```

```
51    if(id == "s") return 83;
52    if(id == "w") return 87;
53  }
54
55
56  // This function sends keys status to server
57  function handleKey(command, down){
58      switch(command){
59      case 37:
60        keys.left = down;
61        break;
62      case 38:
63        keys.forward = down;
64        break;
65      case 39:
66        keys.right = down;
67        break;
68      case 40:
69        keys.backward = down;
70        break;
71      case 65:
72        keys.a = down;
73        break;
74      case 68:
75        keys.d = down;
76        break;
77      case 69:
78        keys.e = down;
79        break;
80      case 87:
81        keys.w = down;
82        break;
83      case 81:
```

```
84        keys.q = down;
85        break;
86      case 83:
87        keys.s = down;
88        break;
89      case 70:
90        keys.f = down;
91        if(localMediaStream) {
92          if(down){
93            record();
94          } else {
95            stop();
96          }
97        }
98        break;
99        default:
100         break;
101     }
102     socket.emit('keypress', keys);
103 }
```

5.2.3 Video

Pri vzpostavitvi socket komunikacije dobi uporabnik naslov, kjer se nahaja video. Uporabnik, ki se poveže preko spleta ima dobi drugačen naslov, kot ima uporabnik, ki se poveže lokalno.

```
1 // Zaledni del
2 io.on('connection', function(socket){
3   // Send default info
4   io.emit('connected', {
5     videoStream: "http://" + ip + ":" +
          videoStreamPort + "/"
6   });
```

```
7  });
8
9  // Uporabniski vmesnik
10 socket.on('connected', function (data) {
11   if(data.videoStream){
12     $("img.background").attr("src", data.videoStream)
           ;
13   }
14 });
```

5.2.4 Zemljevid

Zemljevid Google maps se nahaja v zgornjem desnem kotu uporabniškega vmesnika in je skrit, če še ni pridobil lokacije robota. Knjižnica JavaScript za zemljevid se naloži šele, ko se naloži cela stran, da se izognemo blokiranju in počasnemu nalaganju strani. GPS modul vsako sekundo preko UART vmesnika pošilja NMEA stavke v zaledni del. Zaledni del nato preko socket povezave pošlje samo GPRMC stavek na uporabniški vmesnik. Iz GPRMC stavka razberemo čas, status aktivnosti, zemljepisno širino, zemljepisno dolžino, hitrost, kot v stopinjah, datum ter magnetno variacijo. Zgradba stavka `$GPRMC ,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A` je opisana v tabeli 5.1. Zanima nas samo status, zemljepisna širina in dolžina. Ker so zapisani v obliki minut in sekund, jih moramo pretvoriti v decimalno obliko, da jo lahko uporabimo s knjižnico Google Maps.

V zalednem delu se računalnik Raspberry Pi in modul GPS povežeta preko UART vmesnika. Zaledni del uporabi samo $\$GPRMC$ stavek, ki ga nato preko Socket.io pošlje na uporabniški vmesnik.

```
1 // Zaledni del
2 var serialport = require("serialport");
3 var serialPort = new SerialPort("/dev/ttyAMA0", {
4   baudrate: 9600,
5   parser: serialport.parsers.readline("\n")
```

$GP	Predpona za GPS sprejemnike
RMC	Priporočen minimalni stavek C
123519	Čas sprejema ob 12:35:19 UTC
A	Status A=aktivno ali V=neveljavno
4807.038,N	Zemljepisna šina
01131.000,E	Zemljepisna dolžina
022.4	Hitrost v vozlih
084.4	Kot v stopinjah
230315	Datum sprejema 23.3.2015
003.1,W	Magnetno odstopanje
*6A	Kontrolni podatki

Tabela 5.1: Zgradba GPRMC stavka.

```
6  }, false);
7
8  var SERIAL = {
9    openSerialPort: function(callback){
10     serialPort.open(function (error) {
11       if ( error ) {
12         console.log('Failed to open: '+ error);
13       } else {
14         console.log('GPS connection success');
15         var dataCounter = 0;
16         var gpsData = { gprmc: null };
17         serialPort.on('data', function(data) {
18           if(data.toString().split(",")[0] == "$GPRMC
                ")){
19             gpsData.gprmc = data.toString();
20           }
21
22           dataCounter++;
```

```
23
24            if(dataCounter == 4){
25              callback(gpsData);
26              dataCounter = 0;
27            }
28
29          });
30        }
31      });
32    }
33
34};
35
36module.exports = SERIAL;
37
38
39SERIAL.openSerialPort(function(value){
40  io.emit('gps', value); // Emit gps data
41});
```

Na strani uporabniškega vmesnika najprej zemljevid inicializiramo nato poslušamo na Socket.io povezavi za sporočila iz zalednega dela. Ko se preko povezave Socket.io prejmejo novi GPS podatki, najprej preverimo če so veljavni. Če so veljavni prikažemo zemljevid v zgornjem desnem kotu uporabniškega vmesnika, pretvorimo obliko zapisa in premaknemo lokacijo zemljevida in markerja. Ob primeru, da podatki niso veljavni zemljevid skrijemo.

```
1 // Uporabniski vmesnik
2 var socket = io();
3 var map, marker;
4 var mapLoaded = false;
5 function initialize() {
6   var pos = new google.maps.LatLng(-34.397, 150.644);
```

```
7
8     var mapOptions = {
9       zoom: 18,
10      center: pos,
11      disableDefaultUI: true
12    };
13
14    map = new google.maps.Map(document.getElementById('
          map-canvas'), mapOptions);
15    marker = new google.maps.Marker({
16      position: new google.maps.LatLng(pos),
17      map: map
18    });
19
20    mapLoaded = true;
21  }
22
23  function loadScript() {
24    var script = document.createElement('script');
25    script.type = 'text/javascript';
26    script.src = 'https://maps.googleapis.com/maps/api/
          js?v=3.exp' +
27        '&signed_in=true&callback=initialize';
28    document.body.appendChild(script);
29  }
30
31  window.onload = loadScript;
32
33
34  socket.on('gps', function (msg) {
35    var gprmc = msg.gprmc.split(",");
36    var active = gprmc[2] == "A";
37
```

```
38    if(active){
39      $("#map-canvas").fadeIn();
40    } else {
41      $("#map-canvas").fadeOut();
42    }
43
44    if(map && marker && mapLoaded && active){
45      var lon = convertDegreesMinutesToDecimalDegrees(
          gprmc[3]);
46      var lat = convertDegreesMinutesToDecimalDegrees(
          gprmc[5]);
47
48      var center = new google.maps.LatLng(lon,lat);
49      marker.setPosition(center);
50      map.panTo(center);
51    }
52  });
53
54
55  function convertDegreesMinutesToDecimalDegrees(data){
56    var degrees = parseInt(parseFloat(data)/100);
57
58    if(data.length == 10){
59      var minutes = parseFloat(data.substring(3));
60    } else {
61      var minutes = parseFloat(data.substring(2));
62    }
63
64    var decimalDegrees = parseFloat(degrees + (minutes
        /60)).toFixed(6);
65    return decimalDegrees;
66  };
```

5.2.5 Zajem zvoka

HTML5 ima podporo zajemanja zvoka in videa, ki smo ga uporabili za zajemanje zvoka na uporabniškem vmesniku. RecorderJs [38] je knjižnica za zajemanje in izvažanje zvoka iz spletnih brskalnikov. HTML5 zvočni API dovoli zajemanje zvoka knjižnici RecorderJs, ki ga zajame, ko uporabnik pritisne tipko in izvozi, ko jo spusti. Če ima uporabnik mikrofon, lahko s pritiskom na gumb "F"posname sporočilo, ki ga potem prenesemo preko Socket.io na zaledni del.
Uporabniški vmesnik uporabnika najprej vpraša za dovoljenje uporabe mikrofona. Če uporabnik odobri uporabo, mikrofona se prikaže gumb "F". S pritiskom na gumb "F"knjižnica RecorderJS začne zajemati zvok iz uporabniškega mikrofona. Ob sprostitvi tipke "F"knjižnica RecorderJS ustvari zbirko binarnih podatkov, ki jih pošlje preko povezave Socket.io na zaledni del.

```
1  // Uporabniski vmesnik
2  var localMediaStream = null;
3  var AudioContext = window.AudioContext ||
4                     window.webkitAudioContext ||
5                     window.mozAudioContext ||
6                     window.oAudioContext ||
7                     window.msAudioContext
8  navigator.getMedia = (
9                    navigator.getUserMedia ||
10                   navigator.webkitGetUserMedia ||
11                   navigator.mozGetUserMedia ||
12                   navigator.msGetUserMedia
13 );
14
15 // Request access to MediaStream
16 navigator.getMedia( {audio:true}, function(stream){
17   localMediaStream = stream;
18   audioContext = new AudioContext();
```

```
19    $("#f").fadeIn();
20  }, function(err){
21    console.error("The following error occurred: " +
          err);
22  });
23
24
25  var audioContext;
26  function record() {
27      audioContext = new AudioContext();
28
29      // create a stream source to pass to Recorder.js
30      var mediaStreamSource =
            audioContext.createMediaStreamSource(
            localMediaStream);
31
32      // create new instance of Recorder.js using the
            mediaStreamSource
33      rec = new Recorder(mediaStreamSource, {
34        workerPath: 'recorderjs/recorderWorker.js'
35      });
36
37      // start recording
38      rec.record();
39  };
40
41  function stop() {
42    // stop recording
43    rec.stop();
44
45    // export it to WAV
46    rec.exportWAV(function(value){
47      rec.clear();
```

```
48      socket.emit("audio", value);
49    });
50  }
```

Zaledni del preko Socket.io povezave prejme zbirko binarnih podatkov, ki jih shrani v "voice.wav"datoteko in jo nato predvaja s pomoćjo modula "node-aplay".

```
1  // Zaledni del
2  var express = require('express');
3  var app = express();
4  var http = require('http').Server(app);
5  var io = require('socket.io')(http);
6  var Sound = require('node-aplay');
7  var fs = require('fs');
8  var pwd = "/home/pi/diploma/";
9
10 io.on('connection', function(socket){
11   socket.on('audio', function(data){
12     var buf = new Buffer(data, 'base64');
13     fs.writeFile(pwd + "voice.wav", data, function(
         err) {
14       if(err) {
15         console.log("err", err);
16       } else {
17         console.log("wav created!");
18         new Sound(pwd + "voice.wav").play();
19       }
20     });
21   });
22 });
```

5.2.6 Primer uporabniškega vmesnika

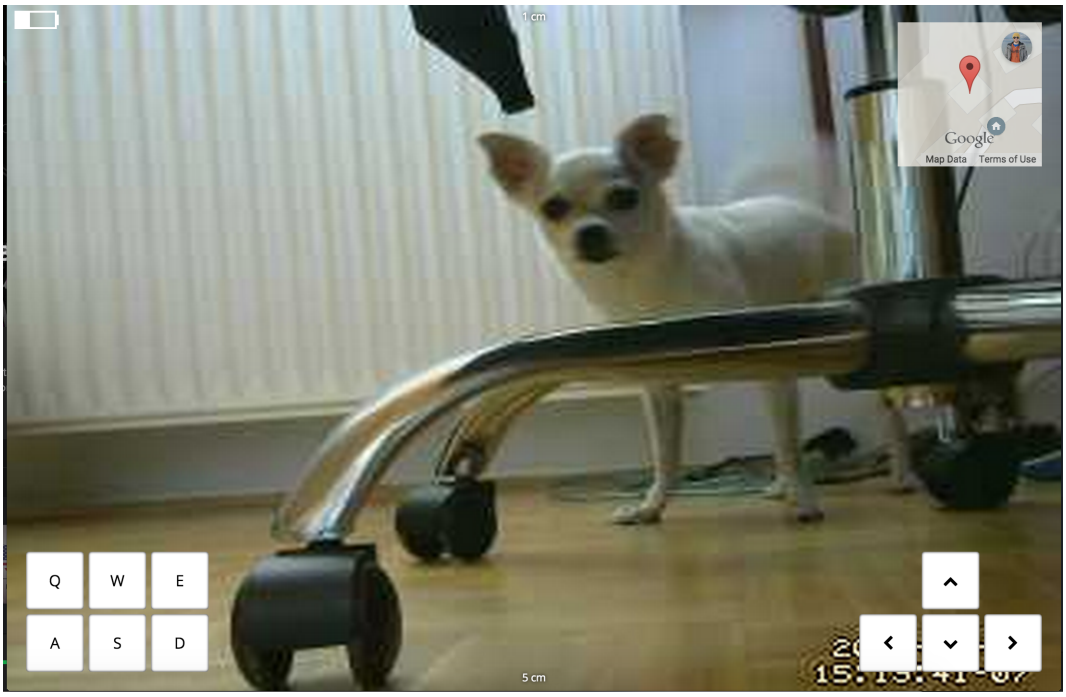
V spodnjem levem kotu se nahajajo gumbi za nadzor nad premikanjem kamere. Kamera se ob vsakem pritisku na gumb pomakne v določeno smer.

- Q - pomakne kamero na začetno pozicijo levo/desno
- E - pomakne kamero na začetno pozicijo gor/dol
- A - pomakne kamero levo
- D - pomakne kamero desno
- W - pomakne kamero navzgor
- S - pomakne kamero navzdol

Spodaj desno se nahajajao gumbi za nadzor nad premikanjem vozila. Z držanjem gumba se bo robot premaknil v izbrano smer.

- Puščica gor - robot se pelje naprej
- Puščica dol - robot se pelje vzvratno
- Puščica levo - robot se obrne levo
- Puščica desno - robot se obrne desno

Če ima uporabnik mikrofon, se prikaže tudi gumb "F", ki deluje na način "pritisni in govori". V desnem zgornjem kotu se od leve proti desni nahajajo indikator za stanje baterije, indikator za kvaliteto brezžične povezave, gumb za pomoč in število povezanih uporabnikov. Grafični prikaz stanja baterije uporabniku enostavno sporoča stanje baterije. Ko je baterija prazna, je priporočljivo robota ugasniti in ponovno napolniti baterijo, da se baterija ne poškoduje. Desno od indikatorja baterije je indikator za kvaliteto brezžične povezave. Gumb za pomoč prikaže okno, kjer so na kratko opisani gumbi na zaslonu. Če sta na robota povezana dva ali več uporabnikov, se bo pojavilo



Slika 5.1: Primer uporabniškega vmesnika

število trenutno povezanih uporabnikov ter ikona, ki predstavlja več uporabnikov. Na vrhu in na dnu uporabniškega vmesnika sta prikazani informaciji o oddaljenosti objekta od robota spredaj in zadaj. Na koncu pa je zgoraj desno še zemljevid, ki prikazuje trenutno lokacijo robota. Zemljevid je viden samo takrat, ko GPS modul lahko pridobi lokacijo. Zato je v zaprtih prostorih običajno zemljevid na uporabniškem vmesniku skrit.

5.3 Zaledni del

5.3.1 Časovne naloge

Zaledni sistem izvaja tri časovne naloge. Prva naloga je branje razdalje iz ultrazvočnih merilnikov razdalje, ki se izvede vsakih 500 milisekund. Branje iz ultrazvočnih merilnikov razdalje bi moralo biti hitrejše, vendar so moduli

prepočasni za hitrejše delovanje. Prekinitev branja modula je 750 milisekund, ko je razdalja do predmeta večja od razdalje, ki jo zmore ultrazvočni merilec razdalje. Druga časovna naloga bere stanje baterije preko SPI, ki komunicira z analogno-digitalnim pretvornikom vsakih 20 sekund. Tretja časovna naloga je preverjanje kvalitete brezžične povezave. Node.js lahko izvaja sistemske ukaze kot je `iwconfig wlan0 | grep -i quality` s katerim lahko izvemo kvaliteto brezžične povezave. Koda s katero beremo razdalje iz ultrazvočnih merilnikov razdalje vsakih 500 milisekund:

```
1 every('500ms').do(function() {
2   // Read and emit ultrasonic range finders data
3   io.emit('message', USONIC.readSensors());
4 });
```

5.3.2 Upravljanje motorjev

Servo modul nadzoruje gonilnik za motorje, ki je nastavljen tako, da z njim komuniciramo kot s servo motorjem. Ko uporabnik pritisne na tipko za premikanja robota na uporabniškem vmesniku, se preko povezave Socket.io sporočijo spremembe v zaledni del. Zaledni del nato spremembe sporoči servo modulu, ki potem ustrezno nastavi širino oddanih pulzov do gonilnika motorjev.

Modul za nadzor servo motorjev nadzoruje tudi servo motorja, ki premikata kamero. Nadzor deluje podobno kot pri nadzoru premikanja, s to razliko, da se ob spremembi servo motor premakne samo za en korak, ki je določen v nastavitvah. Nastavljena so tudi maksimalna razpona pulzov za servomotorja, ki obračata kamero.

Koda Servo modula, ki nadzira servo motorje:

```
1 var piblaster = require('pi-servo-blaster.js');
2 var every = require('schedule').every;
3
4 // Sabertooth accepts pwm ranges from 1000us - 2000
    us.
```

```
5  // piblaster outputs from 500us - 2500us with 10us
       steps.
6
7  var motor1pin = 4;
8  var motor2pin = 25;
9  var middle = 0.15;
10 var addonMotor1 = 0.017;
11 var addonMotor2 = 0.023;
12
13 var servoPitchPin = 17;
14 var servoPitchDefault = 0.11;    // Default position
       of pitch servo
15 var servoPitchPos = servoPitchDefault;
16 var pitchMin = 0.075;
17 var pitchMax = 0.21;
18
19 var servoYawPin = 18;
20 var servoYawDefault = 0.16;      // Default position
       of yaw servo
21 var servoYawPos = servoYawDefault;
22 var yawMin = 0.05;
23 var yawMax = 0.21;
24
25 var servoStep = 0.005;
26
27 var jobSpeed, jobSteer; // Jobs
28 var speed = 50;
29 var steer = 50;
30 var interval = 100; // Intefval of acceleration
31
32
33 var SERVO = {
34   setPitchServo: function(keys){
```

```
35      if(keys.w) { // Up
36        if(servoPitchPos-servoStep >= pitchMin)
             servoPitchPos -= servoStep;
37      }
38
39      if(keys.s) { // Down
40        if(servoPitchPos+servoStep <= pitchMax)
             servoPitchPos += servoStep;
41      }
42
43      if(keys.q) { // Back to default
44        servoPitchPos = servoPitchDefault;
45      }
46
47      if(keys.w || keys.s || keys.q) {
48        console.log("Pitch", servoPitchPos);
49        piblaster.setPwm(servoPitchPin, servoPitchPos);
50      }
51    },
52
53    setPitchServoStick: function(value){
54      value = 100 - value;
55      var pos = ((pitchMax - pitchMin) * value/100) +
           pitchMin;
56      if(value == 50) pos = servoPitchDefault;
57      piblaster.setPwm(servoPitchPin, pos);
58    },
59
60    setYawServo: function(keys){
61      if(keys.a) { // Left
62        if(servoYawPos+servoStep <= yawMax) servoYawPos
             += servoStep;
63      }
```

```
64
65      if(keys.d) { // Right
66        if(servoYawPos-servoStep >= yawMin) servoYawPos
              -= servoStep;
67      }
68
69      if(keys.e) { // Back to start
70        servoYawPos = servoYawDefault;
71      }
72
73      if(keys.a || keys.d || keys.e) {
74        console.log("Yaw", servoYawPos);
75        piblaster.setPwm(servoYawPin, servoYawPos);
76      }
77    },
78
79    setYawServoStick: function(value){
80      var pos = ((yawMax - yawMin) * value/100) +
           yawMin;
81      if(value == 50) pos = servoYawDefault;
82      piblaster.setPwm(servoYawPin, pos);
83    },
84
85    setMotor1: function(value){
86      piblaster.setPwm(motor1pin, value);
87    },
88
89    setMotor2: function(value){
90      piblaster.setPwm(motor2pin, value);
91    },
92
93    setMotorsAccelerate: function(data){
94      // Speed
```

```
95      if(data.forward || data.backward || data.left ||
            data.right){
96        if(data.forward && speed < 60) speed = 60;
97        if(data.backward && speed > 40) speed = 40;
98        if(!data.forward && !data.backward) steer = 50;
99        if(data.left) steer = 60;
100       if(data.right) steer = 40;
101       if(data.left && speed == 50) steer = 70;
102       if(data.right && speed == 50) steer = 30;
103       if(!data.right && !data.left) steer = 50;
104
105       SERVO.setMotorsStick({ x: steer, y: speed });
106
107       if(jobSpeed) jobSpeed.stop();
108       jobSpeed = every('100ms').do(function(){
109         if(speed < 100 && speed > 0) {
110           if(data.forward && speed < 70) speed += 1;
111           if(data.backward && speed > 30) speed -= 1;
112         }
113
114         SERVO.setMotorsStick({ x: steer, y: speed });
115         //console.log(speed, steer);
116       });
117
118     } else {
119       if(jobSpeed) jobSpeed.stop();
120       if(!data.forward && !data.backward) speed = 50;
121       if(!data.left && !data.right) steer = 50;
122       SERVO.setMotorsStick({ x: steer, y: speed });
123     }
124   }
125 };
126
```

```
127
128 module.exports = SERVO;
```

5.3.3 Branje stanja baterije

Popolnoma napolnjena baterija oddaja 12.5 V. Čip MPC3008 [32] za analogno-digitalno pretvorbo pa lahko na vhod sprejme največ 5 V. Zato smo s pomočjo delilnika napetosti izhodno napetost baterije znižali na 5 V.

$$V_{out} = \frac{V_1 \cdot R_2}{R_1 + R_2} \tag{5.1}$$

V enačbo (5.1) za delitev napetosti smo vstavili upora, $R_1 = 1000\ \Omega$ in $R_2 = 680\ \Omega$, da smo znižali vhodno napetost na 5 V. Najmanjša dovoljena napetost baterije je 10.5 V, da se izognemo poškodbi baterije. To pomeni, da je izhodna napetost na delilniku napetosti najmanj 4.25 V. Ćip MCP3008 komunicira z Raspberry Pi preko SPI vmesnika. Za lažjo uporabo smo uporabili knjižnico mpc3008.js [33], ki nam pomaga pri komunikaciji s čipom. Časovna naloga Schedule vsakih 20 s pokliče funkcijo, ki prebere vrednost iz čipa. Pridobljena vrednost iz čipa se nato preračuna v procente preostale baterije, ki jih preko Socket.io pošljemo na uporabniški vmesnik. Knjižnica jQuery nato poskrbi za pravilen izris stanja baterije.

Schedule časovna naloga vsakih 20 s pokliče funkcijo readBatteryLevel iz ADC modula, ki prebere vrednost iz čipa. Vrnjeno vrednost nato pošlje preko Socket.io povezave na uporabniški vmesnik.

```
1 every('20s').do(function() {
2   // Read and emit battery data
3   ADC.readBatteryLevel(function(response){
4     io.emit('battery', { percentage: response });
5   });
6 });
```

ADC modul pomaga pri branju podatkov iz kanala 0 na čipu MCP3008. Pridobljene podatke pretvori v procent preostale baterije in jih vrne z pomočjo povratne funkcije.

```
var Mcp3008 = require('mcp3008.js'),
  adc = new Mcp3008(),
  batteryChannel = 0;

var ADC = {
  readBatteryLevel: function(callback){
    adc.read(batteryChannel, function (value) {
      // Output ranges from 1023 to 0
      // 5V is 1023
      // 0V is 0

      // Get percentage from value
      var percentage = Math.floor((value/1023)*100);

      callback(percentage);
    });
  }
};


module.exports = ADC;
```

Uporabniški vmesnik posluša na "battery"kanalu Socket.io povezave. Ko prejme podatke ustrezno izriše stanja baterije.

```
socket.on('battery', function (data) {
  $(".battery-level").css("width", data.percentage +
     '%');
  $(".battery-value").html(data.percentage + '%');
});
```

5.3.4 Modul GPS

Združenje mornarske elektronike je razvilo standard NMEA [37], ki definira vmesnik med različnimi deli mornarske elektronske opreme. Komunikacija GPS sprejemnika je definirana v tej specifikaciji. Modul vsako sekundo preko UART vmesnika oddaja stavke $GPGGA, $GPGSA, $GPRMC, $GPVTG, ki nam sporočijo podrobnosti in kvaliteto pridobljenih koordinat. Zanima nas samo $GPRMC stavek, ki ga preko Socket.io pošljemo uporabniškemu delu aplikacije.

5.3.5 Predvajanje zvoka

Zvok predvajamo s knjižnico node-aplay, ki omogoča uporablja program aplay. Aplay je enostaven program na linux platformi za predvajanje *Waveform audio format* zvočnih datotek (WAV). Zvok se predvaja ob zagonu robota, povezavi novega uporabnika, odklopu uporabnika ter zvok, ki ga uporabniki posnamejo na uporabniškem vmesniku. Zvok iz uporabniškega vmesnika se prenese v WAV medpomnilniški obliki, ki jo zaledni del shrani v datoteko in predvaja.

5.3.6 Kvaliteta brezžične povezave

Node.js lahko izvrši bash ukaz `iwconfig wlan0 | grep -i quality` v zalednem delu, ki nam sporoči kvaliteto brezžične povezave. Ukaz s pomočjo časovnih nalog izvedemo vsakih 5 s, nato vrnjeno vrednost pošljemo preko Socket.io povezave na uporabniški vmesnik. Na uporabniškem vmesniku knjižnica jQuery poskrbi za ustrezen izris.

Koda s katero preko Socket.io vsake 5 s pošljemo vrednost kvalitete brezžične povezave.

```
1 var every = require('schedule').every;
2 var exec = require('child_process').exec;
3
4 every("5s").do(function(){
```

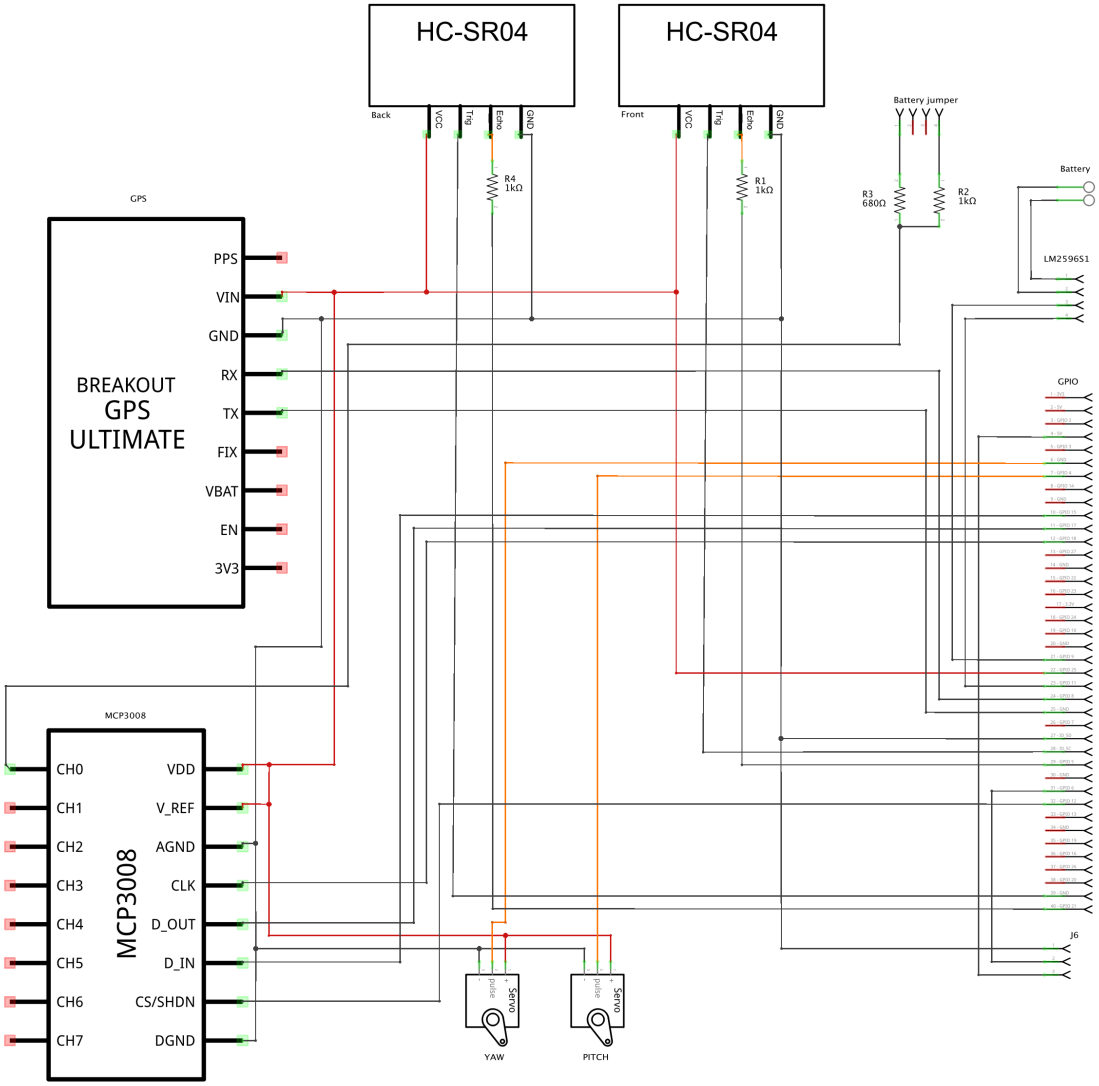
```
5   // Read and emit wifi signal quality
6   exec("iwconfig wlan0 | grep -i quality", function (
        error, stdout) {
7     if (error == null) {
8       try{
9         var quality = stdout.split(" ")[11].split("="
              )[1].split("/")[0];
10        //console.log('Wifi: ' + quality + "%");
11        io.emit('wifi', { quality: quality });
12      } catch(e) {}
13    } else {
14      console.error('Wifi read error: ' + error);
15    }
16  });
17 });
```

5.4 Strojni del

Sprva smo uporabili prototipno ploščico za testiranje vsakega modula posebej. Vsak modul ima svoje specifikacije in navodila za uporabo. Preučiti smo morali vhodne napetosti, porabo energije, način komunikacije in razumevanje prejetih podatkov. Na koncu smo ustvarili še tako imenovan nahrbtnik za Raspberry Pi, ki je kompakten in bolj zanesljiv način kot prototipna ploščica. Slika 5.2 prikazuje shemo elektronsko vezja, ki je bila izdelana s programom Fritzing.

Spisek uporabljenih komponent:

- Računalnik Raspberry Pi 2
- Ultrazvočni merilec razdalje HC-SR04
- Analogno-digitalni pretvornik MCP3008
- Adafruit Ultimate GPS



Slika 5.2: Shema elektronskega vezja.

- USB WIFI dongle TL-WN725N
- Baterija ZIPPY 5000mAh 3S1P 20C Hardcase Pack 11.1v
- Regulator napetosti XL6009E1
- Sabertooth dual 12A
- Logitech HD Webcam C270
- Dagu Wild Thumper 4WD
- MG995 Tower Pro Servo

Poglavje 6

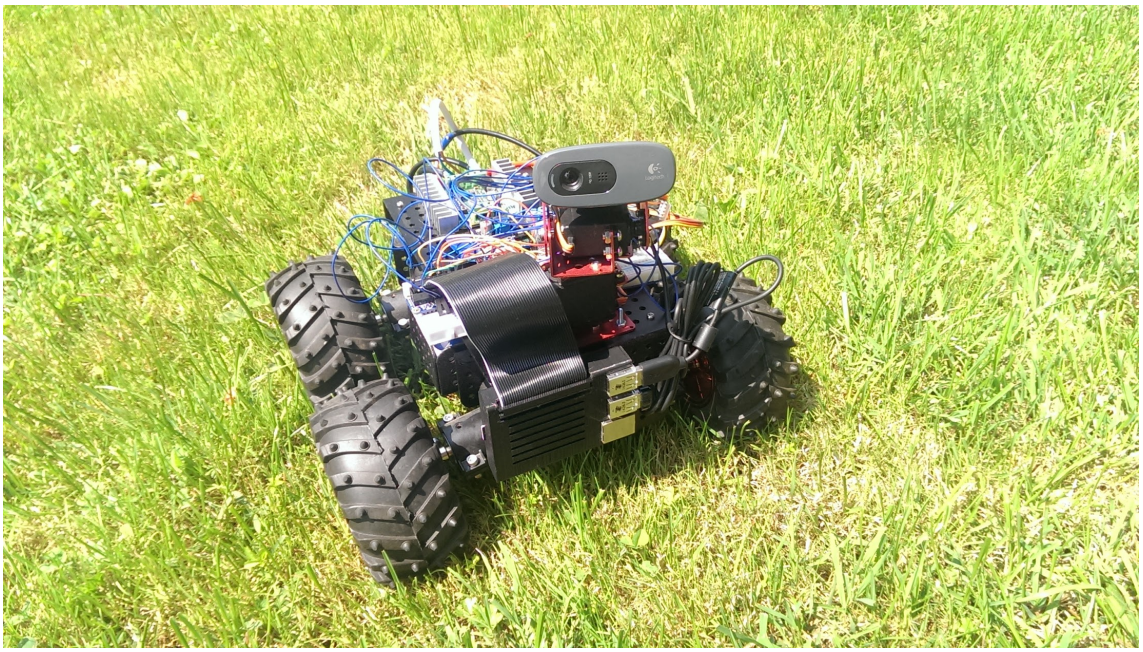
Sklepne ugotovitve

Razvili smo robota z uporabniškim vmesnikom, ki je odziven v realnem času. Uporabnik lahko do uporabniškega vmesnika dostopa preko spletnega brskalnika na lokalnem omrežju. Z odpiranjem vrat za prenos video vsebine na brezžičnem modemu lahko do robota dostopamo tudi preko spleta.

S knjižnicama jQuery in Socket.io smo ustvarili uporabniški vmesnik, ki se v realnem času odziva in spreminja DOM strukturo HTML datoteke. Node.js lahko komunicira z GPIO vmesnikom, izvrši sistemske ukaze in deluje kot strežnik za posredovanje statičnih datotek. Spletne tehnologije se razvijajo v orodje, ki ni namenjeno samo spletnim stranem, ampak tudi namiznim aplikacijam in interakcijo z zunanjim svetom.

Na težave smo naleteli pri napajanju računalnika Raspberry Pi. Zaradi prevelike električne porabe servo motorjev se je Raspberry Pi resetiral. S preobremenitvijo GPIO vmesnika nam je previsoka napetost pokvarila spominsko kartico, zato smo morali ponovno naložiti operacijski sistem.

Projekt je še v zgodnji fazi razvoja. V prihodnosti bo dodan še zvok iz spletne kamere in luč za upravljanje v temnih prostorih. Dodati želimo tudi podporo za USB krmilnik za igralne konzole, ki omogoča boljše upravljanje z robotom. Računalnik Raspberry Pi in senzorje je potrebno v prihodnje tudi zaščititi s po meri natisnjenimi 3D deli. Končni izdelek diplomske naloge lahko vidimo na sliki 6.1.



Slika 6.1: Končni izdelek.

Literatura

[1] HTML [Online]. Dosegljivo:
http://en.wikipedia.org/wiki/HTML. [Dostopano 10. 8. 2015].

[2] HTML5 [Online]. Dosegljivo:
http://en.wikipedia.org/wiki/HTML5. [Dostopano 10. 8. 2015].

[3] CSS [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/Cascading_Style_Sheets. [Dostopano 10. 8. 2015].

[4] SASS [Online]. Dosegljivo:
http://sass-lang.com. [Dostopano 10. 8. 2015].

[5] JavaScript [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/JavaScript. [Dostopano 10. 8. 2015].

[6] Socket.io [Online]. Dosegljivo:
http://socket.io. [Dostopano 11. 8. 2015].

[7] jQuery [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/JQuery. [Dostopano 11. 8. 2015].

[8] Google Maps [Online]. Dosegljivo:
http://en.wikipedia.org/wiki/Google_Maps. [Dostopano 11. 8. 2015].

[9] Node.js [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/Node.js. [Dostopano 12. 8. 2015].

[10] Google V8 [Online]. Dosegljivo: https://en.wikipedia.org/wiki/V8_(JavaScript_engine). [Dostopano 12. 8. 2015].

[11] NPM [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Node.js. [Dostopano 12. 8. 2015].

[12] Express [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Express.js. [Dostopano 12. 8. 2015].

[13] UART [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter. [Dostopano 12. 8. 2015].

[14] SPI [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus. [Dostopano 12. 8. 2015].

[15] Pi-blaster [Online]. Dosegljivo: https://github.com/sarfata/pi-blaster/. [Dostopano 12. 8. 2015].

[16] PWM [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Pulse-width_modulation. [Dostopano 21. 11. 2015].

[17] Schedule [Online]. Dosegljivo: https://github.com/popomore/schedule/. [Dostopano 12. 8. 2015].

[18] Atom [Online]. Dosegljivo: https://atom.io// [Dostopano 12. 8. 2015].

[19] Fritzing [Online]. Dosegljivo: http://fritzing.org/home/. [Dostopano 12. 8. 2015].

[20] Raspberry Pi [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Raspberry_Pi. [Dostopano 13. 8. 2015].

[21] HC-SR04 [Online]. Dosegljivo: http://www.micropik.com/PDF/HCSR04.pdf. [Dostopano 13. 8. 2015].

[22] r-pi-usonic [Online]. Dosegljivo: https://github.com/clebert/r-pi-usonic. [Dostopano 3. 10. 2015].

[23] GPS [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Global_Positioning_System. [Dostopano 13. 8. 2015]

[24] Adafruit Ultimate GPS [Online]. Dosegljivo: https://learn.adafruit.com/adafruit-ultimate-gps. [Dostopano 13. 8. 2015].

[25] Servo motor [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Servomotor. [Dostopano 13. 8. 2015].

[26] How Servo Motors Work [Online]. Dosegljivo: http://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html. [Dostopano 1. 9. 2015].

[27] Logitech C270 [Online]. Dosegljivo: http://www.logitech.com/en-hk/product/hd-webcam-c270. [Dostopano 1. 9. 2015].

[28] Sabertooth [Online]. Dosegljivo: https://www.dimensionengineering.com/products/sabertooth2x12. [Dostopano 13. 8. 2015].

[29] Dagu Wild Thumper 4WD [Online]. Dosegljivo: https://www.pololu.com/product/1567. [Dostopano 1. 9. 2015].

[30] ZIPPY 5000mAh [Online]. Dosegljivo: http://goo.gl/SFhfbJ. [Dostopano 1. 9. 2015].

[31] LiPo [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Lithium_polymer_battery. [Dostopano 13. 8. 2015].

[32] MCP3008 [Online]. Dosegljivo: https://www.adafruit.com/datasheets/MCP3008.pdf. [Dostopano 13. 8. 2015].

[33] mcp3008.js [Online]. Dosegljivo: https://github.com/fiskeben/mcp3008.js/. [Dostopano 5. 9. 2015].

[34] BeagleBone [Online]. Dosegljivo: http://beagleboard.org/bone. [Dostopano 15. 9. 2015].

[35] Raspian [Online]. Dosegljivo: https://www.raspbian.org/. [Dostopano 2. 9. 2015].

[36] Motion [Online]. Dosegljivo: http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome. [Dostopano 15. 8. 2015].

[37] NMEA [Online]. Dosegljivo: http://www.gpsinformation.org/dale/nmea.htm. [Dostopano 13. 8. 2015].

[38] RecorderJS [Online]. Dosegljivo: https://github.com/mattdiamond/Recorderjs. [Dostopano 2. 9. 2015].